### String Operation

<table>
<thead>
<tr>
<th>String Operation</th>
<th>String in Stack</th>
<th>String in Rodata Section</th>
</tr>
</thead>
<tbody>
<tr>
<td>Allocating memory for a string</td>
<td><code>{ char acStr[5]; ... }</code></td>
<td><code>{ ... &quot;hi&quot;... ... }</code></td>
</tr>
<tr>
<td>Initializing a string</td>
<td><code>{ char acStr1[3] = {'h', 'i', '\0'}; char acStr2[] = {'h', 'i', '\0'}; char acStr3[3] = &quot;hi&quot;; char acStr4[] = &quot;hi&quot;; char acStr5[2] = &quot;hi&quot;; /* truncation */ char acStr6[10] = &quot;hi&quot;; ... }</code></td>
<td><code>{ ... &quot;hi&quot;... ... }</code></td>
</tr>
<tr>
<td>Computing the length of a string</td>
<td><code>{ char acStr[10] = &quot;hi&quot;; ... strlen(acStr) ... /* Evaluates to 2 */ ... sizeof(acStr) ... /* Evaluates to 10 */ }</code></td>
<td><code>{ char *pcStr = &quot;hi&quot;; ... strlen(pcStr) ... /* Evaluates to 2 */ ... sizeof(pcStr) ... /* Evaluates to 4 */ }</code></td>
</tr>
<tr>
<td>Changing the characters of a string</td>
<td><code>{ char acStr[10] = &quot;hi&quot;; acStr = &quot;bye&quot;; /* compiletime error */ acStr[0] = 'h'; acStr[1] = 'y'; acStr[2] = 'e'; acStr[3] = '\0'; strcpy(acStr, &quot;bye&quot;); /* Danger of memory corruption. */ }</code></td>
<td>(Runtime error to attempt to change the characters of a string that resides in the rodata section)</td>
</tr>
<tr>
<td>Concatenating characters onto a string</td>
<td><code>{ char acStr[10] = &quot;hi&quot;; acStr += &quot;bye&quot;; /* compiletime error */ acStr[2] = 'h'; acStr[3] = 'y'; acStr[4] = 'e'; acStr[5] = '\0'; strcat(acStr, &quot;bye&quot;); /* Danger of memory corruption. */ }</code></td>
<td>(Runtime error to attempt to change the characters of a string that resides in the rodata section)</td>
</tr>
</tbody>
</table>
### Comparing one string with another

```c
{  char acStr1[] = "hi";
    char acStr2[] = "bye";
    if (acStr1 < acStr2) ...  /* Legal, but compares addresses!!! */
    if (strcmp(acStr1, acStr2) < 0) ...  /* Compares strings */
}
```

(Same as string in stack)

### Reading a string

```c
{  char acStr[10];
    iConvCount = scanf("%s", acStr);
    /* Reads a word as a string.  
       Grave danger of memory corruption. */
    iRet = gets(acStr);
    /* Reads a line as a string,  
       removing the \n character.  
       Grave danger of memory corruption. */
    iRet = fgets(acStr, 10, stdin);
    /* Reads a line as a string,  
       retaining the \n character. */
}
```

(Runtime error to attempt to change the characters of a string that resides in the rodata section)

### Writing a string

```c
{  char acStr[] = "hi";
    iCharCount = printf("%s", acStr);
    /* Writes a string. */
    iSuccessful = puts(acStr);
    /* Writes a string, appending a \n character. */
    iSuccessful = fputs(acStr, stdout);
    /* Writes a string. */
}
```

(Same as string in stack)

### Converting a string to another type

```c
{  char acStr[] = "123";
    int i;
    long l;
    double d;
    iConvCount = sscanf(acStr, "%d", &i);
    i = atoi(acStr);
    l = atol(acStr);
    d = atof(acStr);
}
```

(Same as string in stack)

### Converting another type to a string

```c
{  char acStr[10];
    int i = 123;
    iCharCount = sprintf(acStr, "%d", i);
    /* Danger of memory corruption. */
}
```

(Runtime error to attempt to change the characters of a string that resides in the rodata section)